VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2016 Jan Velčovský

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Velčovský**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ISSA CZECH s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radim Bača, Ph.D.**

Konzultant bakalářské práce: Ing. Michal Kolesár, Ph.D.

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 19. dubna 2016 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 19. dubna 2016 .....

ISSA CZECH s.r.o.
Hrušovská 3203/13a
702 00 Ostrava
DIČ: CZ25381920
www.issa.cz

Rád bych poděkoval vedení a všem spolupracovníkům ze společnosti ISSA Czech s.r.o. za možnost vykonání praxe, spolupráci, získání zkušenosti a znalosti z reálného provozu. Zvláště bych pak rád poděkoval pánům Ing. Pavlu Janáčkovi a Ing. Jaroslavu Smutníkovi, kteří mi předali mnoho užitečných a praktických rad oblasti vývoje systémů. Také vedoucímu mé bakalářské práce panu Ing. Radimu Bačovi za odborné a zkušené rady při tvorbě a zlepšení této písemné zprávy.

Abstrakt

Písemná práce o praxi ve společnosti ISSA Czech s.r.o. a popis zadaných úkolů ve firmě včetně řešení. Hlavním cílem praxe byla spolupráce na vývoji webové aplikaci Ubytovani.net, který má sloužit pro vyhledávání ubytovacích zařízení v České a Slovenské republice s možností filtrace. Součástí praxe bylo také rozšíření a úprava firemního frameworku pro internetové obchody pro nově vznikající e-shopy, včetně jejich tvorby a na stylování.

Klíčová slova: ISSA Czech s.r.o., webová aplikace, informační systém, internetový obchod, LaTeX, bakalářská práce, ubytovani, PHP

Abstract

Bachelor thesis about individual job in the company ISSA Czech s.r.o. with description task and solutions. The aim of practice was a cooperation in a programming of internet application Ubytovani.net. Ubytovani.net is application for searching accommodation in the Czech Republic and Slovakia republic. Next part practice was creating new e-shops and websites

Key Words: ISSA Czech s.r.o., web application, Information System, e-shop, LaTeX, Bachelor thesis, accommodations, PHP

Obsah

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
HTML	–	Hyper Text Markup Language
PDO	–	PHP Data Objects
PHP	–	Personal Home Page
ORM	–	Objektově relační mapování
SQL	–	Structured Query Language
MVC	–	Model View Controller
CMS	–	Content Management System

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Od rozmachu internetu v České republice, každý rok exponenciálně roste zájem o webové stránky a informační systémy. Důvod je prostý, většina uživatelů dnes vyhledává informace právě na internetu. Proto většina společností dnes investuje peníze do webových stránek a internetových reklam. Další podstatnou věcí pro firmy jsou informační systémy, které dokáží ušetřit pracovníkům firem čas a také zefektivnit produktivitu firmy a její růst. Tyto služby a mnohé další nabízí společnost ISSA Czech s.r.o. ve které jsem vykonával praxi.

V rámci praxe jsem se podílel na vývoji webové aplikace určené pro vyhledávání ubytovacích zařízení v České a Slovenské republice. Vzhledem k rozsáhlosti projektu mi byla přidělena práce na veřejné částí systému, což obnášelo programování administrace pro uživatele, rezervačního prostředí a hlavně logiky vyhledávače. Pro stylování prostředí, pak byl použit framework Bootstrap, který umožňuje tvorbu responzivních webových aplikací a stránek.

Další z úkolům praxe byly práce na firemním frameworku pro internetové obchody. Práce obnášela rozšíření frameworku o nové funkce, například o možnost tvorby nových parametrů skrze administraci, nebo tvorbu nových e-shopů.

2 Profil firmy a pracovní zařazení

2.1 Zaměření firmy

Společnost ISSA CZECH s.r.o. byla založe v roce 1997 v Ostravě. Je to menší společnost zabývající se vývojem informačních systému, tvorbou webových aplikací a internetovou bezpečností. Název firmy ISSA představuje zkratku "Internet Software and Security Advice“. Společnost klade vysoký důraz na požadavky zákazníků a nabízí komplexní řešení. Firma poskytuje i další služby v oblasti informačních technologií, jako jsou datová uložiště, massmailing, dohledová centra nebo správa sítí.

Základní rysy firmy:

- Tvorba webových prezentací a internetových aplikací
- Vývoj informačních systémů dle specifických požadavků
- Internetový marketing
- Tvorba internetových obchodů
- Poskytování webhostingu, datových uložišť, emailů
- Poradenství v IT
- Poradenství a realizace v bezpečnosti v IT
- SEO
- Massmailing (newslettry)

2.2 Zařazení studenta

Ve firmě jsem pracoval na pozici programátora. Mým úkolem byly práce na vývoji webových aplikacích jako webové prezentace, internetové obchody či informační systémy, také jsem zpracovával požadavky na úpravy na již běžících webových stránkách. Většinou se jednalo o menší úpravu jako struktury webu, stylování, úpravy obsahů nebo doplnění funkčních požadavků. Součástí praxe byla také komunikace s některými zákazníky, zpracování nových požadavků a prezentace vývoje systému.

3 Seznam úkolu ve firmě

3.1 Vyhledávač Ubytovani.net

- Obecné, migrace dat - str.18
- Administrace pro správce ubytovacích zařízení - ukázka tvorby nové stránky - str.18
- Vyhledávač - str.20

3.2 Firemní framework pro internetové obchody

- Dynamické parametry produktů - str.23
- Rozšíření struktůry obchodu - str.25
- E-shop Praktis.cz - str.27

3.3 Web ITS.cz - str.28

4 Rozbor existujícího řešení

Tato část popisuje existujícího řešení frameworků[1], aby bylo možné pochopit jednotlivé řešení. Část obsahuje také popis některých problému a úkonů, které bylo nutné provést.

4.1 Ubytovani.net

Projekt Ubytovani.net je webová stránka pro vyhledávání ubytovacích zařízení v České a Slovenské republice. Tento web existuje od roku 2003 a prošel vývojem a úpravami mnoha firem. Web umožňuje vyhledávat ubytovací zařízení podle lokace a typu ubytování. Systém za dobu své existence prošel mnoha změnami, nikoliv však modernizací. Úkolem firmy bylo vytvořit nový systém pro tento portál na nových technologiích. Tato možnost umožnila vytvoření nových funkčních vlastností jako rozhraní pro komunikaci mezi zákazníkem a hoteliérem, generování smluv a další.

První fází projektu byla analýza existujícího systému a databáze. Systém ubytovani.net se skládá ze dvou celků. Jedná se o web *zajimavamista.cz* a *ubytovani.net*, tato práce však bude pojednávat pouze o webu *ubytovani.net*.

Pro analýzu databáze bylo nutné projít jednotlivě všechny tabulky a určit jejich obsahovou hodnotu. Původní databáze měla 68 tabulek, s pohledu databází se jedná o středně malou databázi. Analýzou bylo zjištěno, že databáze se nachází ve velmi špatném stavu a migrace dat do nové struktury bude problematická. Jednalo se především o problémy s primárními klíči, které v některých tabulkách zcela chyběly. Dalším ze závažných problému byly datové typy, kde v mnohých tabulkách byly používány pouze typy Varchar i pro číselné hodnoty a chyběla tedy typová kontrola. O návrh nové struktury databáze se postaral firemní specialista na databáze.

4.1.1 Framework Ubytovani.net

Framework pro Ubytovani.net je rozdělen do tří základních adresářů, toto rozdělení má logické opodstatnění.

Adresář *base* obsahuje třídy a skripty pro funkcionalitu jádra systému. Zde jsou umístěny definice třídy pro práci s datovou a kontrolní vrstvou, třídy pro definici základních vlastností o třídy objektů a funkce pro ověřování pravomocí.

Základní třídy

- DBManager - Třída slouží jako vrstva pro komunikaci mezi aplikaci a PDO, funkce třídy zpracuvávají dotazy a pracují s nimi přes transakce
- Portal - Třída definuje základní vlastnosti objektu pro databázový záznam
- PortalPage - Třída představuje základní vlastnosti stránky a kontrolní vrstvy
- Principal - Třída pro ověřování pravomocí přístupu v systému

[1] Framework je soubor knihoven, který vytváří jádro systému a usnadňuje vývoj

- PrintHTML - Třída definující funkce pro generování objektů formulářů a dalších HTML elementů

Adresář *admin* obsahuje skripty administrační části. V tomto adresáři najde složku *custom*, ve které se nachází obsah administrace konkrétního projektu. Jsou zde umístěny skripty pro generování obsahu stránek a podadresář *classes* s třídami reprezentující konkrétní záznamy v databázi a třídy pro kontrolní vrstvu.

Adresář *public* obsahuje skripty a třídy pro funkčnost veřejné části webu. Pro lepší orientaci v systému je zde použita stejná hierarchie adresářů jako v administrační části. Adresář *custom* je rozšířen o adresáře *layout* a *views*, které obsahuji nezbytné skripty pro generování obsahu stránek.

4.1.2 MVC model frameworku Ubytovani.net

Model - datová vrstva

Třídy představující datovou vrstvu a jsou reprezentovány soubory s příponou **.class.php*. Třídy, u kterých se předpokládá práce s databází, dědí z rodičovské třídy *Portal*. Tato třída obsahuje funkce pro práci s třídou *DBManager*, kde jsou definovány funkce pro PDO. Třída také předává svým potomku funkce *Load*, *Delete* a další. Pro každou novou třídu, která používá Potral, musíme nadefinovat funkce *SetFromStatment*, *getSaveData* a *Clear*. Funkce *SetFromStatment* je volána při funkci *Load*, kde získaný záznam z databáze je rozparsován v této funkci a hodnoty jsou nastaveny do jednotlivých proměnných objektu. Funkce *getSaveData* pracuje opačný způsobem, proměnné objektu přiřazuje ke klíčům a vrací výsledek jako pole. Tato funkce je volána nadřazenou funkci *Save* z třídy *Portal*. Pomocí atributu *id* systém určití, zda bude provádět vkládání nebo úpravu existujícího záznamu. Protože většinu základní operací je řešeno pomocí automatiky, je nutné pro atributy, se kterými se pracuje vytvářet gettry a settry.

View - vzhled

View definuje vzhled a strukturu generované stránky. V administrační části jsou definovány dva základní stavy *view*. První stav je označován jako *List*, tato stránka generuje seznam objektů, které představují jednotlivé záznamy v databázi. V Listu lze definovat, které informace se budou vypisovat o objektu a také lze volit filtry, které se definují v kontrolní vrstvě. Druhý stav je označován jako *Detail*, v této časti jsou vypsány všechny proměnné, které lze upravit. *Detail* rozlišuje další stavy *Edit*, *NonEdit* a *New*. Tyto stavy ovlivňují generovaný formulář a události v kontroléru. Pro tvorbu formulářů a jejich vstupních polí se používají funkce třídy PrintHTML, která obsahuje všechny typy polí a pracuje s danou proměnou objektu, funkce navíc rozlišuje tyto tři stavy a není tedy nutno psát kód pro jednotlivé situace. Soubory pro view v administraci jsou umístěny v adresáři *custom*.

Ve veřejné části systému se pro generování obsahu používají třídy typu *layout*. Obsah pro generování není volám ve view, ale používá se kontrolér, který používá danou třídu layout a volá vybrané funkce. Toto řešení umožňuje měnit layout stránky na základě podmínek v kontroléru,

jedná se podobný způsob řešení *stav* bez nutnosti definování nových stavů. Soubory view pro veřejnou část jsou umístěny v adresáři *custom/views*.

Pro spuštění stránky je nutné ve view definovat, kterou stránku a kontrolní vrstvu spouští pomocí funkce *CatalogPage::startPage*, tato funkce má jako jediný parametr název třídy kontroléru stránky.

Controller - kontrolní a logická vrstva

Představuje v systému kontrolní vrstvu pro model a view. Je reprezentován souborem s názvem třídy, které se vztahuje a příponou **page.class.php*. V kontrolérů jsou definovány funkce pro validaci formulářů a funkce na události ve stránce. Aby bylo možné kontrolér využívat musí dědit z nadřazené třídy *PortalPage*. V kontrolní vrstvě mohou být definovány atributy filtrů pro výpis záznamů ve *View*. Tyto atributy začínají klíčovým slovem *filter*.

Události definované v kontroléru je možnost volat přímo ve stránce pomocí formuláře nebo pomocí JS. Pro lepší orientaci v názvoslovích funkcí, začínají funkce pro události klíčovým slovem *do*. Další klíčová slova jsou *before* a *after*, které umožňuji definovat akce, které se vykonají před nebo po dané funkce. Klíčová slova se píšou na začátek názvu funkce, není možná je kombinovat.

Veřejná část systému definuje navíc klíčové slovo *can*. Tato funkce musí být definována, pokud se využívá funkce *do*. Funkce *can* jsou booleovského typu a umožňuji nastavit podmínky, zda se provede volání funkce s prefixem *do*.

4.2 Framework pro e-shopy

Firemní framework pro internetové obchody je založen na stejném rozdělení vrstev (MVC). Framework v současné době používá starší verzi jádra a neumožňuje rozdělení View s použitím tříd *layout* pro generování obsahu stránek. Framework je rozdělen fyzicky do dvou částí administrace systému a obchod, toto řešení umožňuje umístit administraci a web na dva rozdílné servery s rozdílným nastavením. Rozdělením je dosaženo lepší správy a zatížení serveru, kde se předpokládá, že administrace nepotřebuje takový výpočetní výkon jako obchod.

V první části bude popsána struktura administrace. Administrace obsahuje standardně adresáře *classes*, *js*, *css* a *lang*. Jádro administrace je umístěno v adresáři *base*, který se nachází ve složce *classes*. Adresář *base* obsahuje třídy *IdObject*, *PortalPage*, *PrintHTML*, které vytváření jádro systému a pracují s jednotlivými vrstvami. Dle standardů frameworku se vlastní třídy ve frameworku umísťují do adresářů podle jejich zařazení v menu, tedy je-li v administraci v menu položka *Čísleníky*, která má položky *Barva*, *Výrobce*, budou tyto třídy uloženy v adresáři *classes/ciselniky*. Třídy pro práci s datovou a kontrolní vrstvou se umísťují vždy do stejných adresářů. Struktura adresářů pro obchod obsahu *custom*, *classes* a další. Adresář *classes* se musí shodovat adresářem *classes* v administraci, aby byla dodržena kompatibilita datové vrstvy. Adresář *custom* obsahuje všechny skripty a soubory pro konkrétní projekt. Pokud je zapotřebí jakákoliv změna na třídách v adresáři *classes* a je potřebná pouze pro obchod, pak se vytvoří nová třída v adresáři *custom/classes*, která rozšíří danou třídu. Pokud by se pravidla nedodržovala,

mohl by jiný programátor přepsat při aktualizaci všechny úpravy a funkce, které jsou v obchodě používány.

Základní třídy

- IdObject - Třída definuje základní vlastnosti objektu pro databázový záznam a obstarává funkce pro komunikaci s databází

4.2.1 MVC model frameworku pro E-shopy

Vzhledem k podobnosti MVC modelu obou frameworků není nutné popisovat model znovu. Bude tedy následovat část, kde budou popsány rozdíly a popřípadě čím jsou jednotlivé rodičovské třídy nahrazeny.

Model - datová vrstva

Základní konstrukce a vlastnosti jsou identické s modelem frameworku pro Ubytovani.net. Rodičovskou třídu *Portal* nahrazuje *IdObject*. Tato třída definuje funkce pro ukládání, mazání a načítání záznamů. Systém pro ukládání dat používá funkci *Save*, která klíčům pole přiřadí hodnoty proměných a pole předá funkci *InsertOrUpdate*, která vygeneruje a zpracuje daný SQL dotaz.

View - vzhled

Tvorba View v administraci je zcela shodná s VIEW frameworku pro Ubytovani.net. Změna je již při tvorbě VIEW pro obchod. Základní šablonu vzhledu stránky tvoří soubor *layout.php* umístěn v adresáři *custom*. Vzhled jednotlivých stránek jsou definován jednotlivými view soubory, které jsou ukládány v adresáři *custom*.

Controller - kontrolní a logická vrstva

Kontrolní vrstva v této verzi jádra neobsahuje funkce pro definování vzhledu pomocí funkcí *pHeadHtml*, *pContent* a *pFootHtml*. Vzhled stránek lze definovat pouze pomocí *view* souboru a atributu *state*. Oproti novější verzi jádra, postrádá starší řešení bezpečnosti pomocí třídy *Principal*, která před vykonáním funkce umožnovala pomocí funkci s klíčovým slovem *Can* definovat podmínky.

5 Praktická část

Tato část se bude zaobírat vybranými řešeními problému a úkolů u daných projektů.

5.1 Ubytovani.net

5.1.1 Obecné - migrace dat, responzibilita

V teoretické části byla probrána analýza staré databáze, kde bylo zjištěno, že se nachází v nepříznivém stavu. Bylo nutné s klientem probrat, která data se budou migrovat do nové struktury databáze. Protože mnoho sloupců obsahovalo typově nekorektní data, bylo nutné tyto data zahodit a nepřenášet. Jediným řešením bylo ruční doplnění takových to dat přes administrační rozhraní systému. Současná verze systému obsahu na 77 tabulek.

Při migraci dat bylo nezbytné uchovat v tabulce hotel původní *id* záznamu, které je v novém databázi označováno jako *cislo_hotelu*. Tento atribut bylo nutné zachovat z důvodu vyhledávačů a SEO z důvodu URL adres jednotlivých hotelů, kde *id* je klíčový parametr. Původní systém běžel na 13 let a jeho indexace ve vyhledávačích dosahovala nejvyšších pozic.

Většina webů s vyhledávacím potenciálem musí splňovat nároky i pro použití na mobilních zařízeních. Bylo tedy nutné řešit i rensponzibilní verzi pro všechny typy zařízení. Volba padla na technologii *Bootstrap*, která umožňuje rozdělení do 4 vrstev mřížky. Jedná se o framework, který pomocí speciálních názvu umístěných do atributu *class* HTML elementů, umožňuje vytvořit rensponzibilní verzi webu s dodržením grafické koncepce.

5.1.2 Uživatelská administrace

Součástí veřejné části systému pro vyhledávání ubytovacích zařízení, bylo vytvoření vhodného administračního prostředí pro jednotlivé správce zařízení. V této časti, bude popsána tvorba jedné z mnoha stránek administračního prostředí, konkrétně se bude jednat o *Popis hotel*. Tato stránka umožňuje uživateli spravovat popisy hotelu, které jsou rozděleny do kategorií. Pro lepší přehlednost jsou stránky rozděleny podle jazykových mutací. Jednotlivé popisy, které lze vyplnit, jsou umístěny pod sebou a jsou řazeny ve stejném pořadí jako v kartě hotelu. Po zvolení daného popisu k editaci či vložení nového textu se zobrazí ve stránce modální okno s textovým polem. Cílem bylo vytvořit jednoduché a přehledné administrační prostředí i pro osoby méně znalé v oblasti webů.

Pro vytvoření nové stránky bylo zapotřebí vytvořit novou třídu (model) s názvem *HotelPopis*, která je umístěna ve složce *custom/classes/hotel*. Tato třída dědí z nadřazené třídy *UbHotelPopis*, která je umístěné v administrační části, tímto způsobem nemusíme vytvářet všechny atributy a funkce nutné pro funkčnost modelu. Třída *UbHotelPopis* je model popisu hotelů a její objekty reprezentují záznamy tabulky *ub_hotel_popis*. Aby bylo možné získat seznam popisů bylo nutné vytvořit statickou funkci, která vrací kolekci objektů typu *HotelPopis*. Funkce se

nazývá *getPopisAll_byIdHotel* má dva parametry. První parametr označuje id hotelu a druhý parametr označuje vybranou jazykovou mutaci.

```
public static function getPopisAll_byIdHotel($hotelId, $selectLangUser) {
        $like = new HotelPopis();
        $like->clear();
        $like->usePrincipalCond = false;
        $like->useECond = false;
        $like->extraCond = 'AND t.id_hotel=' . $hotelId . ' AND t.id_jazyk=' .
            $selectLangUser;
        return $like->findAll($like, null, 0, 0, 'subLoad');
    }
```

Výpis 1: Statická funkce pro získání všech objektů typu HotelPopis pro daný hotel a jazykovou mutaci v PHP

Pro vyhledání všech záznamu se používá funkce *findAll*. Tato funkce je definována ve třídě *Portal* a má šest parametrů z toho je povinný pouze první. První parametr reprezentuje objekt, který se bude v databázi hledat. Pomocí klíčových atributů *extraCond*, *extraOrder*, *extraFrom* lze specifikovat generovaný dotaz, tyto proměnné definuji podmínky, řazení či spojování tabulek. Pro selekci záznamu z databáze byl v tomto případě použit atribut *extraCond*, kde se zapisuje podmínka v SQL jazyce. Zajímavostí je pátý parametr, který definuje událost, která se provede po nastavení záznamu do objektu. V tomto případě se provádí akce *subLoad*, která provede načtení dat pro sub objektů daného objektu, zde se jedná o kategorii popisu hotelu.

Další fází je vytvoření logické a kontrolní vsrtvy stránky. Ta je reprezentována třídou *HotelPopisPage*, která dědí z rodičovské třídy *PortalPage*. Ve třídě jsou nadefinovány pomocné atributy pro práci se záznamy. Aby byla odezva pro uživatele co nejrychlejší a zatížení databáze co nejmenší, existuje, zde kolekce *kategorieList* do které se při zobrazení stránky načtou objekty. Tato kolekce má zde i význam z hlediska bezpečnosti, při mazání či úpravě záznamu ověřujeme, zda příslušný požadavek o smazání záznamu patří danému hotelu. Ve třídě jsou definovány funkce na události ve stránce, například: Uložit záznam, změna jazykové mutace popisu a další. Pro ukázku je zde popsána funkce pro smazání popisu. Každý popis má ve formuláři tlačítko *Smazat*, po kliku na toto tlačítko se provede požadavek na server, který zpracovává a parsuje posílaná data. Systém zjistí, že byl poslán požadavek na funkci *doDeleteItem* a ověří, zda jsou splněny podmínky pro vykonání události. Ve funkci se ověří existence daného záznamu a jestli jej spravuje daný hotel, pokud ano systém vytvoří objekt a vykoná funkci *delete*. Funkce *delete* provede dotaz na databázi a v případě chyby jí zapíše do logu.

Pro vytvoření struktury stránky je zapotřebí použít třídu typu *layout* s názvem HotelPopisLayout a umístěnou v adresáři *custom/layout/hotel*. V této třídě jsou statické funkce, které

generují obsah a strukturu stránky. Vybraná šablona se volá v kontroléru stránky ve funkci *pContent*, kde můžeme volat libovolný layout na základě zvolených podmínek.

Pro zobrazení obsahu stránky je zapotřebí vytvořit soubor *hotelpopis.php*, který je umístěn do adresáře *views*. Tento soubor obsahuje příkazy pro načtení sessiony a funkcí. Pomocí funkce *role* jsou u jednotlivých view definovány pro jakou skupinu uživatelů jsou určeny, není tedy nutné ověřovat přihlášení uživatele v kontroléru. Poté je pomocí funkce *startPage* spuštěna daná stránka. Pro optimalizaci URL se provádí zápis do souboru *web.php*, který je umístěn v adresáři *public*, zde zápis pro jaké URL se zobrazí daná stránka.

Vzhledem k rozsáhlosti projektu a množství funkcí, jsem vybral pouze ty nejzákladnější body, aby bylo možné pochopit složitost a časovou náročnost daných úkolů.

```
function doDeleteItem() {
        $itemDelete = null;
        if (isset($_REQUEST['itemDelete'])) {
            $itemDelete = $_REQUEST['itemDelete'];
            foreach ($this->popisArr as $popisObj) {
                if ($popisObj->getId() == $itemDelete) {
                    $deletePopis = new HotelPopis();
                    $deletePopis->setId($itemDelete);
                    $deletePopis->delete();
                    break;
                }
            }
        }
        if (!$this->getAjax()) {
            return;
        }
    }
```

Výpis 2: Funkce pro smazání záznamu popisu

5.1.3 Vyhledávač

Vzhled k účelu toho webu, byl důraz kladen na vývoj vyhledávače a jeho funkci. Požadavky na funkčnost a vlastnosti se v průběhu vývoje několikrát změnily a budou se měnit pravděpodobně i v budoucnu. V první fázi vývoje byla nutná konzultace se zákazníkem, s kterým bylo dohodnuto, která pole se budou ve vyhledávači vyskytovat. Rozhraní vyhledávače umožňuje vyhledávat ubytování podle *kraje*, *lokality*, *hory* či *tématického ubytování*. Rozšířený vyhledávač umožňuje specifikovat dále *typ ubytování*, *okres* a *rekreační oblast*. Podle původního zadání měl vyhledávač pro web být obdobou původního systému. Původní vyhledávač umožňoval pouze

filtraci podle jedné kategorie a nebyla možná žádná sub filtrace další kategorií, avšak v průběhu vývoj se koncepce vyhledávače změnila a bylo nutné navrhnout řešení, které vytvoří vazby mezi jednotlivými kategoriemi a umožní filtraci podkategorií. V této části nastal problém, protože neexistovala žádná přímá vazba mezi jednotlivými kategoriemi. Byla tedy nutná analýza databáze a možnosti řešení. Nabídla se tedy dvě řešení, první se zakládalo na koncepci vytvoření vazebních tabulek a doplnění vazeb mezi kategoriemi. Toto řešení by bylo z hlediska optimalizace databází ideální, avšak pohltilo by mnoho dní práce. Druhým řešením bylo využití již existujících vazeb a použití záznamů hotelu jako prostředníka, který by vazby na jednotlivé kategorie umožnil provázat. Toto řešení již není z hlediska databází ideální a pokud by neexistoval jediný hotel s propojení s danou část, kategorie se ve vyhledávači nezobrazí. Toto řešení bylo nakonec uskutečněno a vyžádalo si jen několika hodinovou práci s úpravou SQL dotazů. Vyhledávač je úzce spojen s technologií JS. Tato technologie definuje chování vyhledávače a provádí úkony na základě podnětů od uživatele. Především se jedná o změny vybraných kategorií a volání funkci pro aktualizaci obsahu vyhledávače (kategorií). Pokud uživatel změní *kraj*, který je v hierarchii vyhledávače nejobecnější, musí se provést filtrace ostatních kategorií. Tato událost se řeší pomocí JS a AJAXu. Pomocí technologie AJAX se na pozadí stránky provede požadavek na server. Skripty, které zajišťuje zpracování AJAX požadavků se nacházejí v souboru *portal.ajax.php*. Po odeslání požadavků jsou zjištěny povinné parametry, pokud jsou splněny všechny podmínky je pomocí funkce pro větvení *switch* vybrán konkrétní požadavek (sekvence úkonů), který vrací výsledek. Výsledky funkcí určené pro vyhledávač jsou vráceny ve formátu JSON, který umožňuje strukturování dat. V programovacím jazyce PHP se pro převedení dat do formátu JSON používá funkce *json_encode*. Vracený výsledek ve formátu JSON se zpracovává pomocí jQuery knihoven je vytvářen DOM struktura jednotlivých polí vyhledávače. Aby byla možnost nastavit vyhledávač do výchozího stavu, existuje tzv. "Košík", ten vymaže všechna dosavadní nastavení vyhledávače a načte všechny možnosti pomocí AJAX funkcí. Výhodou řešení obsahu vyhledávače přes JS a AJAX je, že není nutné načítání celé stránky, které by zabíralo mnoho času a nepůsobilo na uživatele celistvě.

Po odeslání požadavku na vyhledání zařízení nás systém přesměruje na stránku *search*. Ta má definovanou kontrolní vrstvu *SearchPage*, kde se nachází funkce, která spravuje požadavky na vyhledání dle zadaných kritérií. Funkce *getParametersFromPathInfo* rozlišuje, zda se jedná o fulltextové vyhledávání nebo hledání dle vybraných parametrů, dle toho těchto podmínek je generován SQL dotaz.

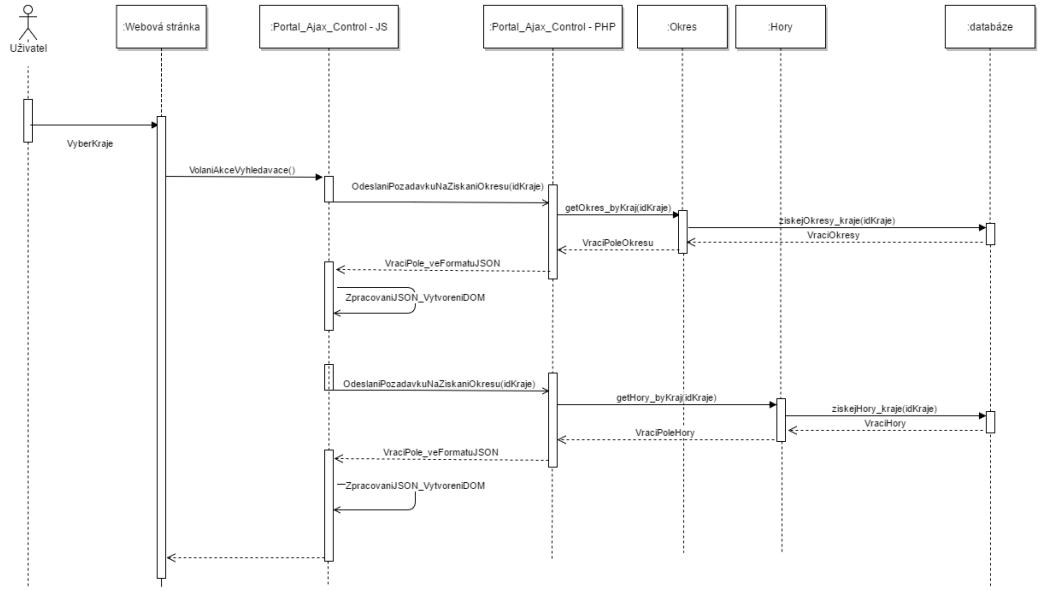
Kontrolní vrstva definuji i funkci pro vytváření titulku stránky podle vyhledávaných hodnot.

Po odeslání požadavků na vyhledání zařízení dle zadaných kritérií se požadavek zpracovává standartním způsobem pomocí kontrolní vrstvy. O zpracování se stará kontrolní vrstva *SearchPage*. V této vrstvě jsou definovány funkce pro generování SQL dotazů, dle zadaných hodnot ve vyhledávači.

Součástí fulltextového vyhledávače je našeptávač možností výběru na základě zadaného textu. Našeptávač vypisuje výsledky do jednotlivých skupin jako *kraj*, *město*, *název hotelu* a

další. Při řešení této častí byla snaha implentace našeptávače frameworku Bootstrap. Webová stranka pracuje s nejnovější verzi Bootstrapu 3.generace, která technologií našeptávače postrádá oproti 2.generaci frameworku. Bylo tedy nutné ověřit zpětnou kompatibilitu frameworku a pokusit se plug-in implementovat ručně. Podle neoficiálních informací byla implementace možná, avšak po její zakomponování nebylo našeptávač využívat dle požadavků zákazníka. Bylo tedy nutné zvolit náhradní řešení s možností využití jiné technologie nebo vytvoření vlastního skriptu pro našeptávač. Na internetu bylo k dispozici mnoho externích skriptů, které by umožnily vytvoření responzivního našeptávače, problém ale vždy nastal s kompatibilitou knihoven pro externí skript a knihovny Bootstrapu, proto nebyla jiná cesta než vytvoření vlastního skriptu s podporou daných verzí jQuery knihoven.

Ukázka průběhu sub filtrace položek vyhledávače při zvolení Kraje. Pokud uživatel zvolí kraj provede se událost, která volá JS funkci na zpracování sub filtrace kategorií. Ve funkci se získá hodnota zvoleného kraje a aktuálně zvoleného státu. Poté se pomocí asynchronní komunikace zažádá o získání položek konkrétní kategorie (okres, hora, lokalita at.). Tyto požadavky se zpracovávají v PHP skriptu, který pomocí klíčových parametrů rozpozná o jaký se jedná požadavek a kterou třídu má o něj žádat. Výsledek hledání se zakóduje do formátu JSON, který je pro další zpracování v JS vhodnější. Poté se ve JS skriptu výsledek rozparsuje a vytvořím nový seznam položek pro volbu. Ukázka sekvečního diagramu ukazuje subfiltraci pouze okresu a hory, protože postup procesuje shodny pro všechny kategorie.



Obrázek 1: Sekveční diagram Vyhledávače - sub filtrace podle kraje

5.2 Framework pro internetové obchody

V rámci vykonávané praxe bylo nutné se seznámit s firemním frameworkem pro internetové obchody, abych byl schopen vykonávat zadané změny na systému a vytvářet nové obchody.

5.2.1 Dynamické parametry

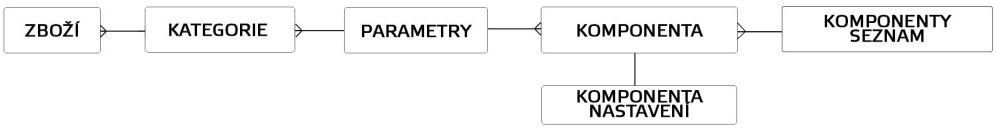
Dosavadní verze frameworku neumožňovala uživatelům vytvářet nové parametry jednotlivým produktům, vždy byl nutný zásah programátora do databáze a MCV modelu. Bylo tedy nutné navrhnout vhodné řešení a poté jej implementovat. Byla tedy nutná i analýza databáze než se s návrhem řešení začalo.

První z navrhovaných řešení bylo navazovat jednotlivé parametry na konkrétní zboží. Toto řešení by bylo ideální za podmínek, že parametr by byl u všech produktů. Pokud by byl definován jen pro několik produktů vznikaly by v tabulce prázdné sloupce a pozdější práce s parametry jako filtry by byla časově náročná.

Druhé řešení bylo vytvářet skupiny parametrů a provazovat je s konkrétním produktem. Toto řešení umožňovalo přiřazovat na zboží jen konkrétní skupinu parametrů. Každá skupiny by měla definovanou vlastní tabulku a atributy, což by minimalizovalo obsah prázdných hodnot v tabulkách. Nevýhodu řešení je však filtrace zboží v rámci kategorie.

A poslední řešení bylo koncepce založená na vytváření skupin parametrů. Skupiny by nenavazovaly na konkrétní produkty, ale na kategorie. Toto řešení eliminuje rozdílnost parametrů u zboží v rámci kategorie a umožňuje filtrovat zboží. Nevýhodou řešení je pokud uživatel vytvoří atribut, který je pouze pro jeden produkt v rámci kategorie.

Volba padla na poslední navrhované řešení. Nyní bylo nutné vytvořit návrh tabulek a jejich vazeb.



Obrázek 2: ER mode návrhu řešení

Komponenty seznam

Objekty této třídy představuji druhy komponent jako číslo, text či reálné číslo. Systém v současné verzi nabízí pouze tři komponenty, ale je jej možné rozšířit i o další jako datum či checkbox.

Komponenta

Objekty této třídy představuji konkrétní komponentu pro danou skupinu. U objektů nastavujeme atributy *nameComponent*, *position*, *parameters* a *componentDefiniton*. Atribut *nameComponent* představuje název komponenty ve výpisu seznamu komponent, ale také představuje název sloupce v tabulce. Atribut *parameters* je objekt třídy *Parametry*.

Komponenta nastavení

Atributy této třídy reprezentují nastavení objekt třídy Komponenta. Jsou to atributy titulek, délka obsahu a filtr, poslední parametr umožňuje nastavit, zda se obsah komponenty bude součástí filtru.

Parametry - Model

Třída *Parametry* definuje datovou strukturu objektů. Obsahuje tři vlastní atributy název tabulky název tabulky parametrů, pořadí a název skupiny. Pro práci s databází jsou definovány funkce, které generují SQL dotazy na vytvoření nebo smazání tabulek. Aby bylo možné pomocí parametru nalézt komponenty existuje funkce *findComponents*, tato funkce vrací kolekci komponent, které jsou ve skupině.

Parametry – Kontrolér a logická vrstva

Logická vrstva definuje úkony na události ve stránce, jako je vyvolání formuláře pro přidání komponenty nebo smazání komponenty. Vrstva obsahuje funkce pro kontrolu validace při událostech. Protože objekty třídy *Komponenta* nemají vlastní kontrolní vrstvu, ale pracuje se s nimi přímo, jsou jejich funkce pro validaci definovány v tomto kontroléru. Součástí vrstvy jsou i funkce pro práci s tabulkou v databázi, které generují SQL dotazy pro práci se sloupci tabulky.

Parametry - View

Soubor obsahuje kód pro generování obsahu. Pomocí atributu *State* jsou definovány stavy, podle niž je vybrán obsah, který se bude generovat. Pro generování prvků formuláře se používá třída *PrintHTML*.

Po vytvoření jednotlivých částí modulu podle navržené koncepce zbývalo upravit modul kategorie pro připojení modulu *Parametry*. Ve třídě *Kategorie* bylo zapotřebí vytvořit nový atribut *parameter*, který je objektem třídy *Parametry*. Aby bylo možné využívat automatických funkcí frameworku při zpracovávání formulářů je nutné pro atribut vytvořit getter, setter a zavést nový atribut do stávajících funkcí *Clear*, *Save* a *SetFromStatement*. Aby uživatelé mohli zvolit

skupinu, bylo nutné upravit View soubor *kategorie.php* a přidat do něj možnost volby skupiny parametrů.

5.2.2 Rozšíření struktury

Dosavadní Framework fungoval na principu dvou úrovní stromové struktury. A však nově vznikající eshop, měl dle přání zákazníka 4-úrovně struktury. Bylo tedy nutné udělat takové úpravy, aby mohl e-shop na frameworku vzniknout. Mým úkolem bylo navrhnout řešení toho problémů a následně jej implementovat.

Byla provedena analýza jak v rámci systému funguje struktura a existuji vazby mezi jednotlivými uzly. Výsledkem analýzy bylo zjištěno, že hlediska ukládání struktury nebude třeba provádět změny. Jediné změny, které bude nutné provést je úprava funkcionalit.

V prvé řadě zapořebí vytvořit podmínky, které musí rodičovský uzel splnit, aby na něj bylo možno navázat potomka. První podmínka byla, že daný uzel může být rodičovský pokud neexistuje žádná přímá vazba se zbožím. Druhou podmínka byla, že uzel nesmí být typu *ViewType*. V rámci kategorií můžeme rozlišovat dva typy *ViewType* a *DataType*. *ViewType* je typ kategorie, která se chová jako kopie vybrané kategorie, není jí možné vybrat v seznamu kategorií a přebírá všechny produkty z vybrané kategorie. Kategorie typu *DataType* je standartní kategorie, u které lze definovat skupinu a lze k ní přiřadit produkty.

Dále bylo zapotřebí vytvořit funkci pro volbu rodičovských kategorií. Funkce se nazývá *printKategorieSelectLastNodeOfStructure* a vrací seznam objektů typu *Kategorie*. Pomocí parametrů funkce můžeme definovat podmínky pro filtraci výsledku.Pokud je definován parametr *masterCategory*, tak výsledkem je seznam kategorií, které mohou být rodičovským uzlem. Pokud není je vrácen seznam kategorií, které jsou v rámci struktury koncovými uzly a nejsou typu *ViewType*, tato funkce je využívána pro výběr kategorií u produktů.

Další krom vývoje bylo vytvoření nových funkcí pro generování menu na stránkách obchodu. Pro výpis celé struktury menu existuje funkce *printMenuStructure*. Funkce v prvním kroku získá všechny kategorie, které nemají rodičovský prvek. Tento seznam se prochází pomocí cyklu a na každý prvkem se volá funkce *printSubItems*. Funkce *printSubItems* ověřuje, zda daná kategorie má potomky, pokud ano, získá pole těchto kategorií a provadí rekurivně funkci *printSubItems*.

```
static function printMenuStructure($page, $rootCategory=false, $showRoot=true)
    {
      global $mm, $http_path;
      loadClass('com/issa/eshop/EshopKategorie');
         $items=array();
      if($rootCategory) {
            $items[]=$rootCategory;
      } else {
        $like = new EshopKategorie();
        $like->setFilterZobrazit(2);
        $items = $like->findLike($like, 0, false, array('all'));
      }
      $idSelectKategorie = (get_class($page)=='EshopZboziPage' ? $page->
          getKategorie()->getId():0);
      echo  ($showRoot ? '<ul>' : '');
         foreach ($items as $key => $item) {
        if($showRoot)
                  echo '<li'.($idSelectKategorie == $item->getId() ? ' class="
                      selected"' : '').'><a href="'.$item->getUrl().'" ><span>'.
                      $item->getNazev().'</span></a>';
          echo '<span class="title-menu">'.$item->getNazev().'</span>';
              MenuCategories::printSubItems($item,$idSelectKategorie);
          echo  ($showRoot ? '</li>' : '');
        }
      echo  ($showRoot ? '</ul>' : '');
}
```

Výpis 3: Funkce pro tisk menu

5.2.3 Praktis

Praktis.cz je internetový obchod, který běžel na informačním systému Magento, který byl náročný na provoz a jakékoliv požadavky na změnu byly náročné. Proto bylo mým úkolem vytvořit nový e-shop na ISSA Frameworku. V prvotní fázi bylo nutné získat specifikace od zákazníka, jaké informace budeme o produktech uchovávat, jaké budou platební podmínky a metody, poplatky a další specifikace. Podle těchto informací byly provedeny nezbytné úkony na frameworku, aby byly splněny požadavky zákazníka. Součástí vývoje byla úprava stromové struktůry, která je popsána výše. Dle dodaných grafických podkladů a požadavků jsem vytvořil rozhraní obchodu. Pro stylování webu byly použity kaskádové styly a jQuery knihovny. Soubor s kaskádovými styly je uložen v adresáři *custom/css*.

Protože tento obchod je velice obsáhly bylo k fulltextovému vyhledávači přidán našeptávač produktů. Při tvorbě návrhnu našeptávače jsem volil mezi tvorbou vlastního kódu nebo použití již existujících řešení. Volba padla na použití knihovny *jQuery Autocomplete*, toto řešení používá asynchronní komunikaci a možnost vlastní úpravy generovaného výsledku, dalším důvod použití knihovny byl, že není zapotřebí vytváření vlastních podmínek pro chování okna.

Funkce našeptávače se nazývá *searchAutocomplete* je umístěna v souboru *eshop.js*. Pomocí AJAX komunikace je odesílán požadavek na PHP skript, který zpracovává hledaný výraz a vytváří SQL dotaz na databázi. Hledaný výraz může být kód zboží, název či pouze část názvu produktu. PHP skript poté vrací pole, kde je strukturován výsledek SQL dotazu. Toto pole je následně procházeno v JS a hodnoty se přiřazují proměným, které vytváření obsah, který se bude zobrazovat uživateli na stránce.

5.3 Webové stránky - ITS.cz

Součástí praxe byla i tvorba webový prezentací na firemním redakčním systému. Protože v dnešní době většina uživatelů prochází weby i na mobilních zařízeních je nutné vytvářet webové stránky responzivní. Toto řešení má výhodu v tom, že obsah webu bude čitelný na všech zařízeních, ale také přináší úskalí při programování takového webové prezentace, pokud má složitější grafický návrh. Výhodou responzivních webů je i ta, že z hlediska SEO jsou takové webové prezentace u vyhledávačů upřednostňovány.

Podle předlohy grafiky bylo za úkol vytvořit nový responzivní web, který by nahradil dosavadní webovou prezentaci. Pro tvorbu responzivního webu byla použita technologie Bootstrap, kterou podporuje i CMS společnosti. Pomocí kaskádových stylů bylo vytvořeno grafické prostředí dle grafického návrhu. Pro vytvoření hlavního menu webu, byla použita i technologie jQuery, aby bylo dosaženo různého chování pro určité velikosti displeje na zařízeních. Pokud je web zobrazen na velkém zařízení, s největší pravděpodobností počítač, tak je hlavní menu zobrazeno a při najetí na položku menu se zobrazí podkategorie menu, pokud existují. Pro zařízení s rozlišením menším než 992px se poté zobrazuje tvz. burger menu. Toto menu je ve výchozím stavu skryté, aby uživatel viděl na první pohled obsah stránky, pokud bude chtít menu zobrazit, provede to kliknutím na blok menu. Menu se poté rozbalí a překryje obsah stránky, pomocí piktogramů může uživatel rozlišit, zda položka v menu má podkategorie nebo zda se jedná o odkaz na stránku. Přechody zobrazování a skrývání položek byly doplněny o animace, aby web působil na uživatele plynule a moderně.

```
function showSubMenu(elem) {
    $("#burgerMenu LI UL").slideUp("slow");
    var selectElem = $(elem).next("UL");
    if (saveLastItemParrent === selectElem.parent().attr('id')) {
        selectElem.slideUp("slow");
        saveLastItemParrent = null;
        $(elem).stop().animate({backgroundColor: '#0E232E'});
        $(elem).removeClass("has_arrow_invert");
    } else {
        selectElem.slideDown("slow");
        $(elem).addClass("has_arrow_invert");
        $("#"+saveLastItemParrent+">A").removeClass("has_arrow_invert");
        $("#"+saveLastItemParrent+">A").stop().animate({backgroundColor: '#0
            E232E'});
        saveLastItemParrent = selectElem.parent().attr('id');
        $(elem).stop().animate({backgroundColor: '#364E59'});
    }
    return false;
}
```

Výpis 4: Funkce pro zobrazení a skrytí podkategorií v burger menu

6 Závěr práce

6.1 Dovednosti uplatněné v průběhu praxe

V průběhu praxe jsem využil znalosti a zkušenosti z předmětu databázových systémů a programování. Schopnosti a znalosti z databázových systému jsem uplatnil při návrhu databáze a tvorbě SQL dotazů pro dílčí částí projektů, zejména pak znalosti dotazovacího jazyk jsem využil při práci na projektu Ubytovani.net. Také jsem využil znalosti z předmětu vývoje informačních systémů, kde jsem si mohl vyzkoušet postup vývoje systému v praxi. Při tvorbě webových stránek na firemním CMS jsem uplatnil své znalosti a zkušenosti pro tvorbu struktur a stylování pomocí kaskádových stylů a znalosti z HTML5 z předmětu vývoje internetových aplikací.

6.2 Získané zkušenosti a znalosti během praxe

Díky možnosti praxe jsem se mohl zdokonalit a naučit nových technologií a řešení při vývoji. Naučil jsem se novým technologiím pro tvorbu responzivních webových stránek a aplikací pomocí technologií Boostrap. Získal jsem větší jistotu pří návrhnu řešení a důvěru ve své schopnosti při implementaci požadavků. Také jsem si osvojil práci se systémem CVS, který se ve firmě používá u většiny projektů. Přínosem byla účast na schůzkách se zákazníky a získání základní znalosti pro práci s nimi.

6.3 Zhodnocení praxe

Za celé období praxe jsem plnil různé zadání od návrhu databáze, úpravu obsahů stránek, vzhledu či vytvoření zcela nových webů až po programování funkčních částí aplikací. Možnost absolvování bakalářské praxe byl obrovský přínos zkušeností, znalostí a hlavně získání představy o reálném pracovním životě. Osvojil jsem si nové schopnosti v organizaci práce a zlepšil znalosti v oboru vývoje webů a informačních systémů. Vyzkoušel jsem si práci v týmu pod vedením zkušenějších kolegů, kdy jsem pracoval na částech projektů a musel organizovat práci s ostatními. Zúčastnil jsem se také schůzek se zákazníky, kdy jsem prezentoval jednotlivé části vyvíjeného systému a konzultoval jsem s nimi další požadavky nebo získával potřebné informace pro vývoj.

Literatura

[1] Citace o firmě *ISSA* [online]. 2016 [cit. 30.1.2016]. Dostupné z: http://issa.cz/o-nas-620.html

[2] Ubytovani.net - Analýza a návrh *Pavel Janaček* [cit. 11.3.2016].

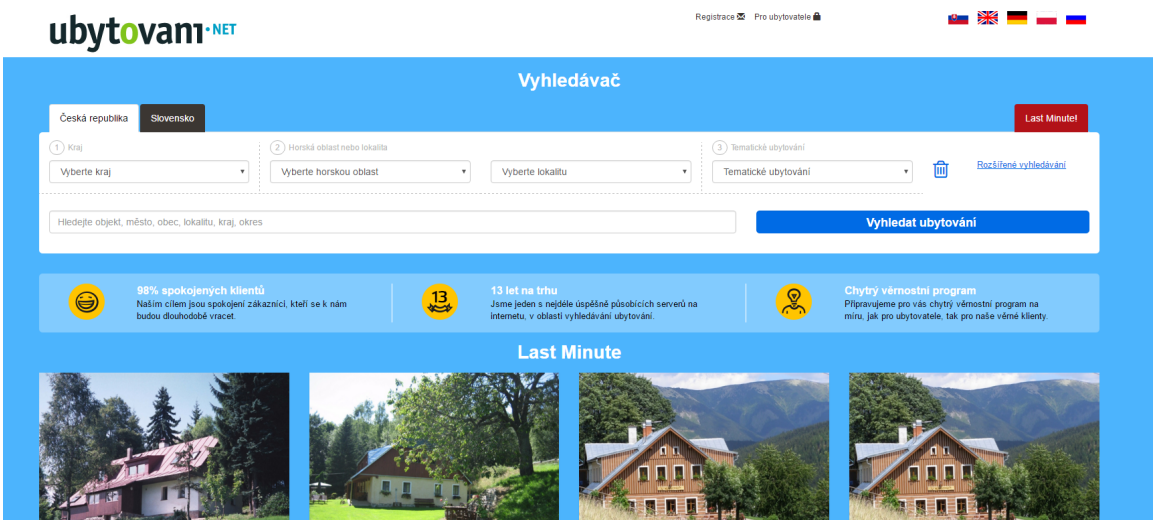
[3] Co je to Framework? *Peter Láng* [online]. 2010 [cit. 31.1.2016]. Dostupné z: http://langi.cz/webarna/co-je-to-framework

[4] Autocomplete *jQuery Api* [online]. Dostupné z: https://jqueryui.com/autocomplete/

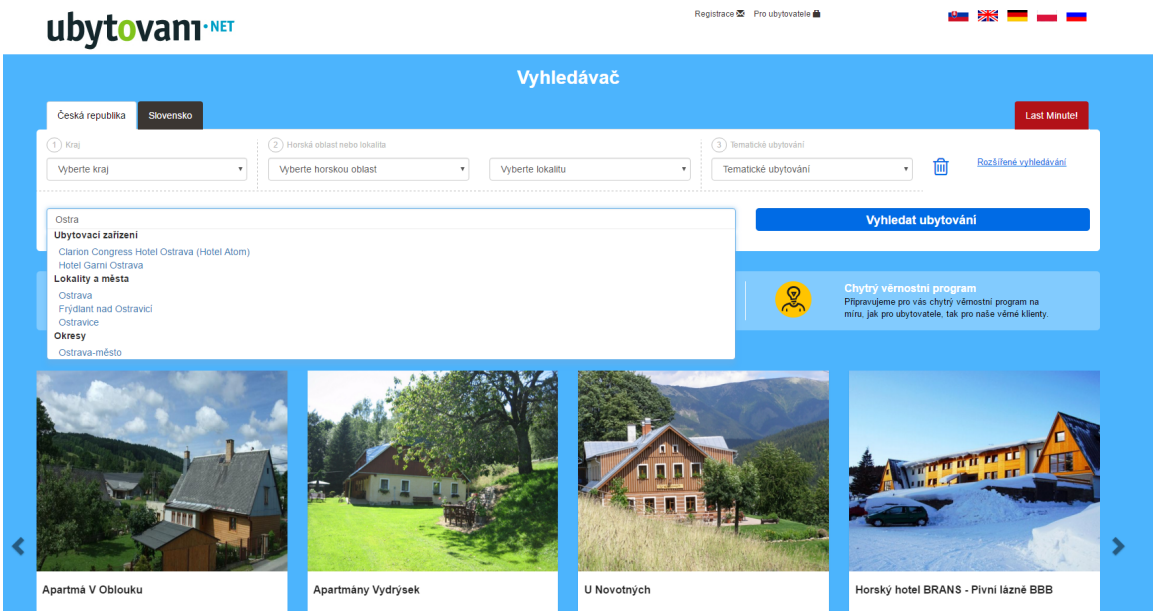
[5] Co je to PDO? *The PHP Group* [online]. Dostupné z: http://php.net/manual/en/intro.pdo.php

7 Obrázky

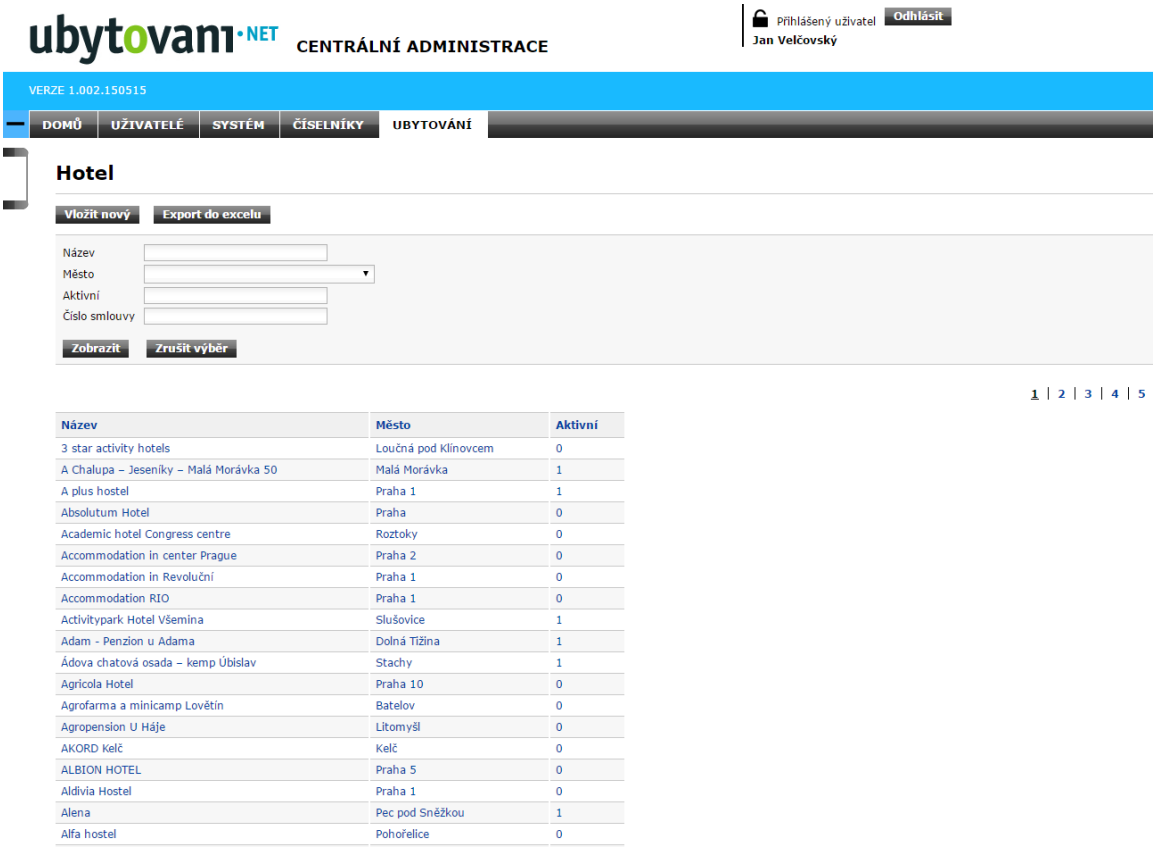
Obrázek 3: Ukázka původního systému - Ubytovani.net



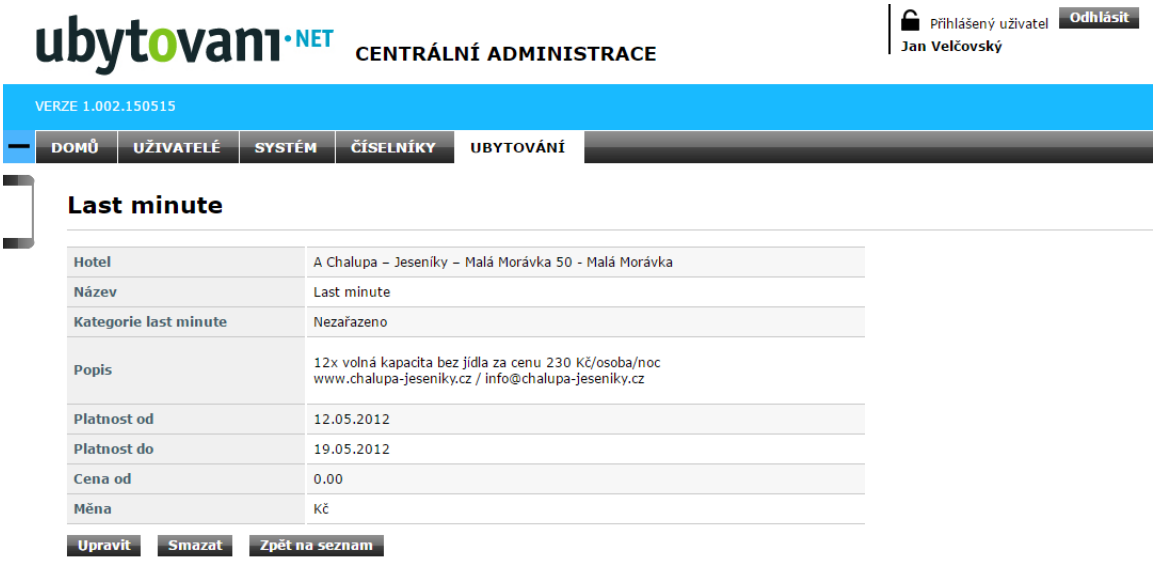
Obrázek 4: Ukázka nové systému - Ubytovani.net



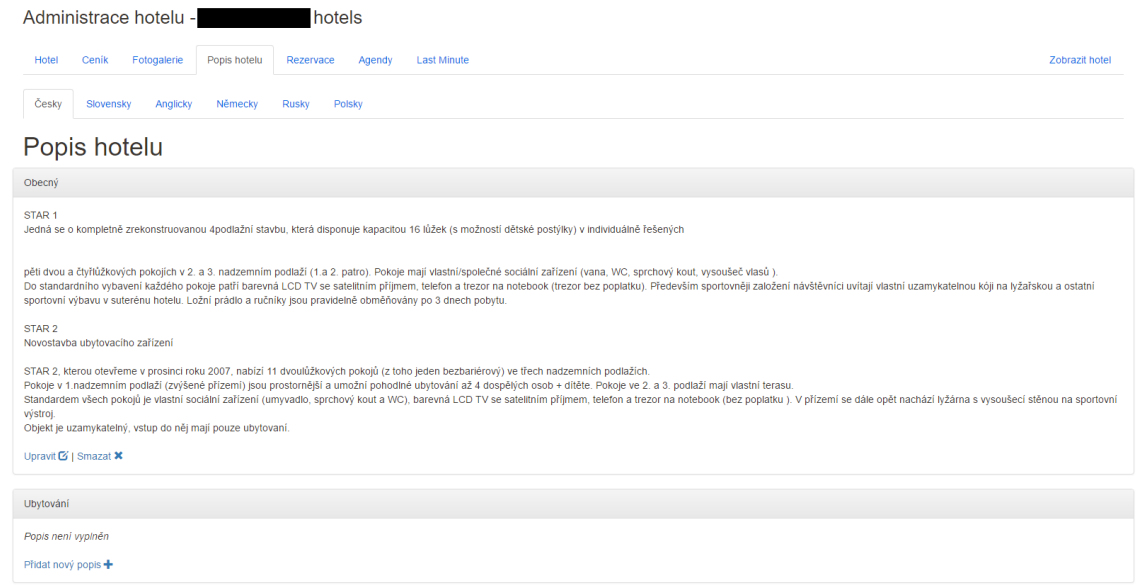
Obrázek 5: Ukázka nové systému - Ubytovani.net



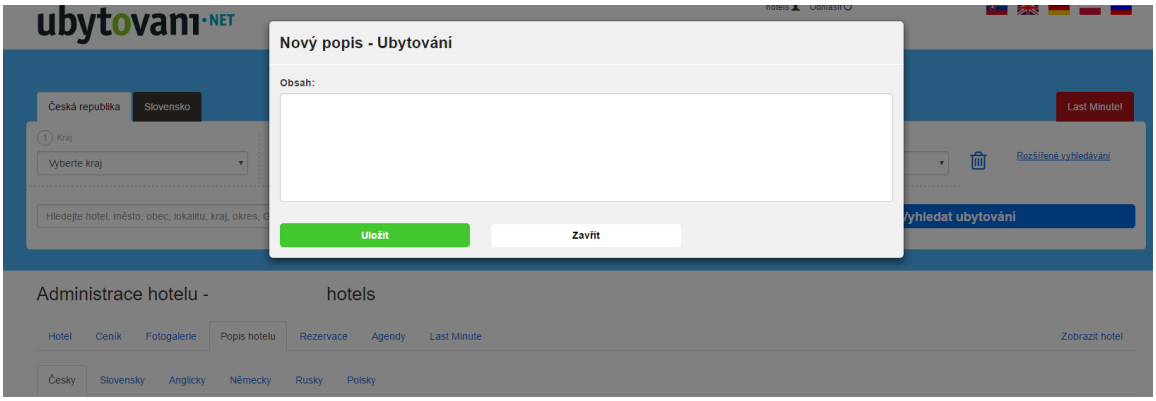
Obrázek 6: Ukázka administrace systému - Ubytovani.net



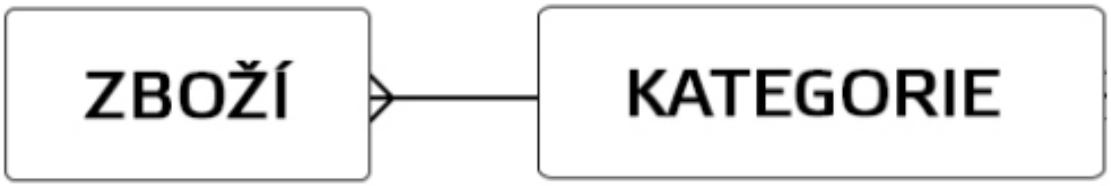
Obrázek 7: Ukázka administrace systému - Ubytovani.net



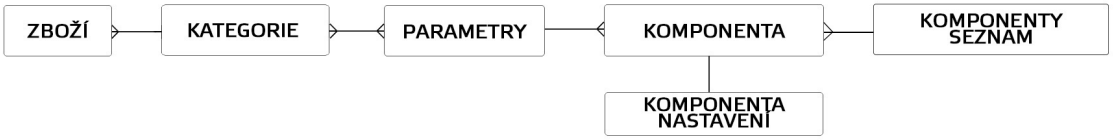
Obrázek 8: Ukázka administrace pro hoteliéra - Ubytovani.net



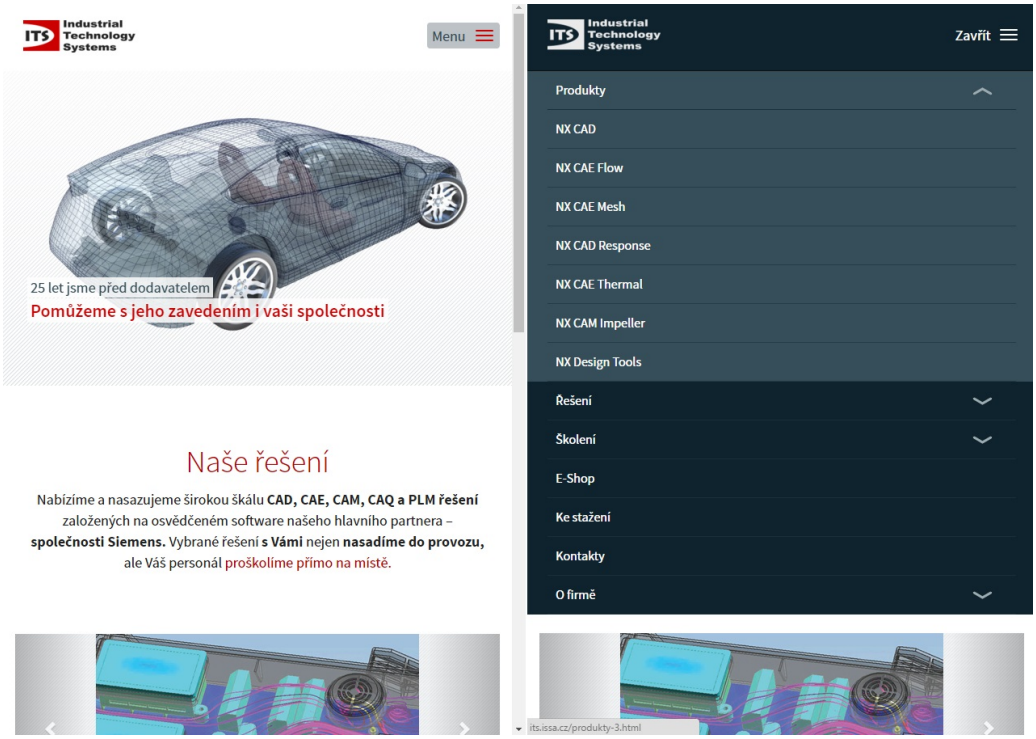
Obrázek 9: Ukázka modálního okna s formulářem - Ubytovani.net



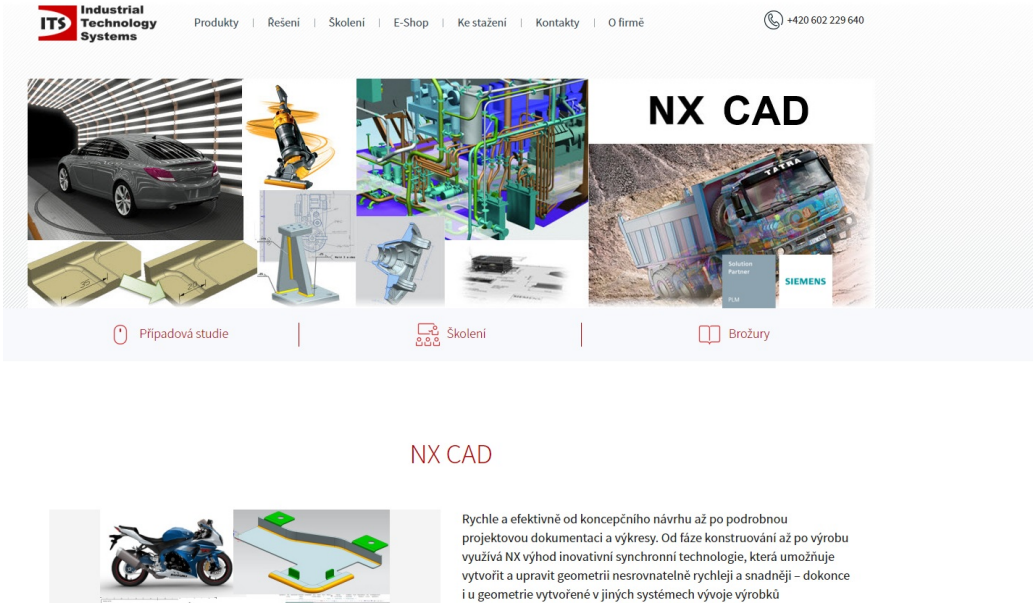
Obrázek 10: Ukázka původní vazby mezi kategorií a zbožím



Obrázek 11: Ukázka současné vazby mezi kategorií, parametry a zbožím



Obrázek 12: Ukázka burger menu webu its.cz - leva část zavřene menu - pravá otevřene menu



Obrázek 13: Ukázka webu its.cz